Claims

1.-6. (Canceled)

7. (Previously presented) In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting one or more in-line code annotations at one or more annotation targets in source code;

wherein the one or more annotations comprise at least one annotation on a first pointer to a buffer, wherein the at least one annotation comprises a property that indicates a characteristic of the buffer, wherein the property that indicates the characteristic of the buffer takes a size argument, and wherein the size argument comprises a location of a second pointer associated with the buffer; and wherein the annotations on the first pointer are placed in an argument list to a function call that uses the first pointer as a parameter.

8.-13. (Canceled)

14. (Previously presented) The method of claim 7 wherein the characteristic is a readable extent of the buffer.

15. (Previously presented) The method of claim 7 wherein the characteristic is a writable extent of the buffer.

16.-20. (Canceled)

21. (Previously presented) The method of claim 7 wherein the at least one annotation includes an annotation prefix.

22.-23. (Canceled)

24. (Previously presented) In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting an annotation at a first value having a first value type in the computer program code;

wherein the annotation comprises a first instance of a keyword, the first instance of the keyword indicating that the first value satisfies all usability properties necessary to allow a first function to rely on the first value, wherein other instances of the keyword identical to the first instance are operable to indicate that other values having different respective value types satisfy all usability properties necessary to allow functions to rely on the respective other values, wherein use of the keyword associates a pre-determined set of usability properties with a value type, and wherein the usability properties depend on the value type.

25. (Canceled)

26. (Canceled)

27. (Previously presented) The method of claim 24 wherein the first value type is scalar, void, pointer, user-defined type, or struct.

28. (Previously presented) The method of claim 24 wherein the first value is a reference parameter.

29. (Previously presented) The method of claim 24 wherein the first value is a pointer, wherein an object pointed to by the pointer has one or more readable elements, the one or more readable elements of the object each having usability properties sufficient to allow the first function to rely on the one or more readable elements.

30. (Currently Amended) In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is

operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting an annotation having an argument comprising a second value type in the computer program code, wherein the annotation annotates a ~~value~~ variable having a first ~~declared~~ value type ~~with a first set of annotation-specific usability properties~~;

wherein the annotation ~~overrides the first set of annotation-specific usability properties of~~ changes the first ~~declared~~ value type ~~and indicates that the value has annotation-specific usability properties that depend on annotation-specific properties~~ of [[a]] the variable to the second value type ~~denoted by the argument~~ of the annotation argument.

31. (Original) The method of claim 30 wherein the first value type is a legacy value type.

32. (Currently Amended) The method of claim 30 wherein the ~~first value type is void * and wherein the~~ second value type has a null-termination characteristic.

33. (Canceled)

34. (Previously presented) In a computer system, a method of annotating computer-executable program code stored on a computer-readable medium, the method comprising:

adding an annotation to a pointer in the computer program code, wherein the annotation describes transferring buffer properties from a second pointer to the pointer; and

including a location parameter with the annotation, wherein the location parameter describes the logical buffer pointed to by the pointer.

35.-40. (Canceled)

41. (Previously presented) The method of claim 7 further comprising using the location of the second pointer associated with the buffer to determine the size of the buffer.

42. (Previously presented) The method of claim 7 wherein the second pointer associated with the buffer is an end pointer for the buffer.

43. (Previously presented) The method of claim 7 wherein the second pointer associated with the buffer is an internal pointer for the buffer.

44. (Previously presented) The method of claim 14 further comprising using the location of the second pointer associated with the buffer to determine the readable extent of the buffer.

45. (Previously presented) The method of claim 15 further comprising using the location of the second pointer associated with the buffer to determine the writable extent of the buffer.

46. (Previously presented) The method of claim 24 wherein the annotation further comprises an except qualifier.

47. (Canceled)

48. (Canceled)

49. (Canceled)

50. (Canceled)

51. (Canceled)

52. (Canceled)

53. (Previously presented) A method of processing annotated computer program code stored on a computer-readable medium, the method comprising:

reading at least one annotation having an argument from the annotated computer program code, wherein the at least one annotation annotates a value having a first declared value type with a first set of usability properties used only in an annotation context, and wherein the annotation overrides the first set of usability properties used only in the annotation context of the first declared value type and indicates a second set of usability properties used only in the annotation context for the value that depend on the second value type denoted by the argument of the annotation, such that the second set of usability properties are used in the context of a second annotation rather than the first set of usability properties;

processing the annotated computer program code based at least in part on the second set of usability properties for the value; and

outputting a result of the processing.

54. (Previously presented) The method of claim 53 wherein the second value type is a null-terminated string type.

55. (Previously presented) The method of claim 53 wherein the processing comprises determining whether the value satisfies the second set of usability properties.

56. (Previously presented) The method of claim 24 wherein if the second value type is a pointer, the usability properties necessary to allow the first function to rely on the first value comprises the pointer pointing to a buffer with at least one readable element.

57. (Previously presented) The method of claim 24 wherein if the first value type is a scalar, the usability properties necessary to allow the first function to rely on the first value comprises the scalar being initialized.

58. (Previously presented) The method of claim 53 wherein if the first value is a pointer, the second set of usability properties comprises the pointer pointing to a buffer with at least one readable element.

59. (Previously presented) The method of claim 53 wherein if the first value is a scalar, the second set of usability properties comprises the scalar being initialized.

60. (Previously presented) The method of claim 53 wherein if the first value is a struct, the second set of usability properties comprises each field of the struct being initialized.

61. (New) The method of claim 30 wherein the second value type must be a visible type at program point where the annotation is placed.